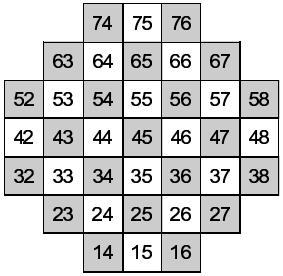
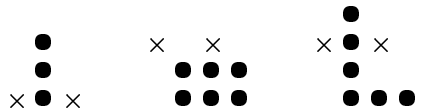
Declarative Action Programs

- Agent Logic Programs
- Declarative Semantics
- Procedural Semantics
- Sensing

Example: Peg Solitaire



An Agent Logic Program (ALP)



3-pattern 6-pattern L-pattern

```
is_pattern([C,X1,X2,X3]) :-?(peg(C)),
                           X1 = C+1, X2 = X1+10, X3 = X1+20,
                           ?(peg(X1) and peg(X2) and peg(X3)).
is_pattern([C,X1,X2,X3]) :-?(peg(C)),
                           X1 = C-1, X2 = X1+10, X3 = X1+20,
                           ?(peg(X1) and peg(X2) and peg(X3)).
...
```


ALP for Peg Solitaire (Cont'd)

```
board_solved :- ?(peg(45) and forall(X, X=45 or not peg(X))).

pattern_solved(P) :- member(Y,P), do(jump(X,Y,Z)),
                     pattern_solved(P).
pattern_solved([Catalyst|P]) :- ?(peg(Catalyst)), empty(P).

empty([]).
empty(X|L]) :- ?(not peg(X)), empty(L).

strategy :- board_solved.
strategy :- is_pattern(P), pattern_solved(P), strategy.
```

ALP Syntax

- Signature includes all terms of sort FLUENT and ACTION
- If `p` is an *n*-ary relation and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is a program atom
- If *a* is an ACTION term, then `do`(*a*) is a program atom
- If ϕ is a state property composed of FLUENT terms, then ?(ϕ) is a program atom
- If $H, B_1, \ldots, B_n$ are program atoms, then $H :- B_1, \ldots, B_n$ is an ALP clause
- An ALP is a finite set of ALP clauses
- An ALP query is finite sequence of program atoms

Declarative Semantics

Clause and Query Expansion

For a clause $H :- B_1, \ldots, B_n$ let $s_1, \ldots, s_{n+1}$ be a sequence of TIME variables.

- For $i = 1, \ldots, n$
  - If B_i is of the form $p(t_1, \ldots, t_m)$, it is expanded to $p(t_1, \ldots, t_m, s_i, s_{i+1})$;
  - If B_i is of the form `do`(*a*), it is expanded to *Poss*(*a*, s_i, s_{i+1});
  - If B_i is of the form ?(ϕ), it is expanded to $\phi[s_i]$ and $s_{i+1} = s_i$.
- The head $H = p(t_1, \ldots, t_m)$ is expanded to $p(t_1, \ldots, t_m, s_1, s_{n+1})$.
- `:-` is replaced by $\subset$ and "`,`" is replaced by "$\wedge$".
- A query is expanded like the body of a clause but with s_1 set to the initial TIME constant.

Example

```
strategy :- board_solved.
strategy :- is_pattern(P), pattern_solved(P), strategy.
```

is expanded to

Strategy(s_1, s_2) $\subset$ *BoardSolved*(s_1, s_2)

Strategy(s_1, s_4) $\subset$ *IsPattern*(p, s_1, s_2) $\wedge$ *PatternSolved*(p, s_2, s_3) $\wedge$ *Strategy*(s_3, s_4)

```
board_solved :- ?(peg(45) and forall(X, X=45 or not peg(X))).
```

is expanded to

BoardSolved(s, s) $\subset$ *Holds*(*Peg*(45), s) $\wedge$ ($\forall x$) ($x = 45 \vee \neg$*Holds*(*Peg*(x), s))

Importance of Atom Ordering

```
p :- ?(f), do(a).
```

is expanded to

$P(s_1, s_2) \subset Holds(F, s_1) \wedge Poss(A, s_1, s_2)$

```
p :- do(a), ?(f).
```

is expanded to

$P(s_1, s_2) \subset Poss(A, s_1, s_2) \wedge Holds(F, s_2)$

Requirements for Time Structure (1)

Action execution in ALPs is strictly sequential.

Therefore the time structure must satisfy

$$Poss(a, s, t) \supset s < t$$

$$Poss(a, s, t) \wedge Poss(a', s', t') \supset (t < t' \supset t \leq s') \wedge (t = t' \supset a = a' \wedge s = s')$$

Note: This is always true if situations are used.

Requirements for Time Structure (2)

There are no "gaps" in between action executions.

Therefore the time structure must satisfy

$$(\exists t)\ Poss(a, s, t) \wedge (\forall f)\ (Holds(f, s) \equiv Holds(f, s')) \supset (\exists t')\ Poss(a, s', t')$$

Note: This is always true if situations are used.

Example for "Gaps"

`p :- do(a), do(b)` is expanded to

$P(s_1, s_3) \subset Poss(A, s_1, s_2) \wedge Poss(B, s_2, s_3)$

With the precondition axioms

$$Poss(A, s, t) \equiv s = 0 \wedge t = 1$$

$$Poss(B, s, t) \equiv s = 2 \wedge t = 3$$

there is no derivation for query `p`

Procedural Semantics

Derivation Rules (1)

- Standard atoms:

$$\frac{Q_1 \wedge Q_2 \wedge ... \wedge Q_n}{(B_1 \wedge ... \wedge B_m \wedge Q_2 \wedge ... \wedge Q_n)\theta}$$

where Q_1 is a literal defined in the ALP and $H \subset B_1 \wedge ... \wedge B_m$ is the expansion of a clause in the ALP such that $Q_1\theta = H\theta$, for some substitution θ.

Derivation Rules (2)

- Actions:

$$\frac{Poss(a, s, t) \wedge Q_2 \wedge ... \wedge Q_n}{(Q_2 \wedge ... \wedge Q_n)\theta}$$

where $\sum \models Poss(a, s, t)\theta$ with substitution θ on the variables in $Poss(a, s, t)$.

- Tests:

$$\frac{\phi[s] \wedge Q_2 \wedge ... \wedge Q_n}{(Q_2 \wedge ... \wedge Q_n)\theta}$$

where $\sum \models \phi[s]\theta$ with substitution θ on the variables in ϕ.

Example

```
office(alice, 101).
office(bob,   102).
deliver :- ?(has_package_for(P)), office(P,R), do(go(R)).
```

The expansion is

Office(*Alice*, 101, *s*, *s*)

Office(*Bob*, 102, *s*, *s*)

$Deliver(s_1, s_3) \subset Holds(HasPackageFor(p), s_1) \wedge Office(p, r, s_1, s_2) \wedge$
$Poss(Go(r), s_2, s_3)$

A Successful Derivation

$Deliver(S_0, s)$

$Holds(HasPackageFor(p), S_0) \wedge Office(p, r, S_0, s_2) \wedge Poss(Go(r), s_2, s)$

$Office(Bob, r, S_0, s_2) \wedge Poss(Go(r), s_2, s)$

$Poss(Go(102), S_0, s)$

□

(where we assume the precondition axiom $Poss(Go(r), s, t) \equiv t = Do(Go(r), s)$.)

Soundness

If there exists a successful derivation for Q with computer answer θ then the domain axiomatization and the expanded program together entail $Q\theta$, provided the reasoner for the domain axiomatization is correct.

Incompleteness

```
p :- ?(f).
p :- ?(not f).
```

The expansion

$P(s, s) \subset Holds(F, s)$

$P(s, s) \subset \neg Holds(F, s)$

entails $(\exists s)\ P(S_0, s)$ but there is no derivation for query `p`.

An ALP Interpreter

- See the lecture notes on how FLUX can be used as an interpreter for ALPs.
- See the lecture notes on how state update axioms can be considered a special case of general effect axioms.

ALPs with Sensing

Histories

A history *h* is a sequence of (action, sensing result)-pairs

$(a_1, v_1), (a_2, v_2), ..., (a_n, v_n) \quad (n \geq 0)$

- $End[h, s] :<=> Do(a_n, ..., Do(a_2, Do(a_1, s)) ...)$
- $Sensed[h, s] :<=> \{\neg\} SF(a_1, Do(a_1, s)) \wedge ... \wedge \{\neg\} SF(a_n, End[h, s])$

  where "¬" is placed wrt. a_i iff $v_i = false$

Generalized Derivation Rules

- Actions:

$$\frac{Poss(a, s_1, s_2) \wedge Q_2 \wedge ... \wedge Q_n}{(Q_2 \wedge ... \wedge Q_n)\theta}$$

  where $\sum \cup \{Sensed[h, s]\} \models Poss(a, s_1, s_2)\theta$ with substitution θ on the variables in $Poss(a, s_1, s_2)$.

- Tests:

$$\frac{\phi[s] \wedge Q_2 \wedge ... \wedge Q_n}{(Q_2 \wedge ... \wedge Q_n)\theta}$$

  where $\sum \cup \{Sensed[h, s]\} \models \phi[s]\theta$ with substitution θ on the variables in ϕ.